Isabelle

By tkloht

May 3, 2016

Contents

1 Example theory file for getting acquainted with Isabelle

1.1 Terms

We can write logical formulae and terms in the usual notation. Connectives such as $\neg$,$\vee$, $\wedge$ etc. can be typed using the backslash \ followed by the name of the sign. I.e. *not* for $\neg$. Note that during typing *not* at some point there will be a pop-up menu offering you certain auto completion suggestions that you can accept by pressing the tab key.

1.2 Types

All terms (and also constant symbols, variables etc.) are associated a type. The type *bool* is the type of all Boolean-values objects (e.g. truth values). New types can be inserted at will.

typedecl *i* — Create a new type i for the type of individuals

1.3 Constants

New constants can be defined using the *consts* keyword. You need to specify the type of the constant explicitly.

1.4 Terms and Formulas

In higher-order logic (HOL), terms are all well-formed expressions that can be expressed within the logic. A term has a unique type, such as in $f\ A$ where the term $f\ A$ has type i. Terms of type *bool* are called "formulas".

1.4.1 Example formula 1

If it's raining the street will get wet

consts *raining* :: *bool* — constant symbol for raining
consts *wet* :: $i \Rightarrow$ *bool* — predicate symbol for wet
consts *street* :: *i* — constant symbol for the street

prop $raining \longrightarrow wet(street)$ — raining implies street-is-wet

1.4.2 Example formula 2

consts *good* :: $i \Rightarrow$ *bool* — predicate symbol for being good

prop $good(A)$ — A is good

A is a free variable of the above term, hence it is not closed

1.4.3 Example formula 3

prop $\forall A.\ good(A)$ — everything is good

A is a a bound variable of the above term, which is universally qualified.

1.5 Proofs

We will learn how to formalize proofs in Isabelle throughout this course.

1.5.1 Proofs with handy keywords

theorem *MyFirstTheorem*:
  assumes A
  shows $B \longrightarrow A$
proof $-$
  {
    assume B
     from *assms* **have** A **by** $-$ — Iterate the fact that A holds by assumptions using the - sign
  }
  then have $B \longrightarrow A$ **by** (*rule impI*)
  thus *?thesis* **.**
qed

1.5.2 Proofs with labels

theorem *MyFirstTheorem2*:
  assumes *1*: A
  shows $B \longrightarrow A$
proof $-$
  {
    assume B
    from *1* **have** A **by** $-$
  } **note** *2* = *this*
  from *2* **have** $B \longrightarrow A$ **by** (*rule impI*)
  thus *?thesis* **.**
qed

1.5.3 Using the proofs

We can now derive simple facts of the above theorem.

corollary *ThatFollowsDirectly*:
  assumes A
  shows $P(A) \longrightarrow A$
using *assms* **by** (*rule MyFirstTheorem*[**where** $B = P(A)$])

theorem *excludedMiddle*:
  shows $A \vee \neg A$

proof −
  {
    assume *1*: $\neg(A \vee \neg A)$
    {
      assume *2*: $\neg A$
      from *2* **have** *3*: $A \vee \neg A$ **by** (*rule disjI2*)
      from *1* **have** *4*: $\neg(A \vee \neg A)$ **by** −
      from *4 3* **have** *False* **by** (*rule notE*)
    }
    from *this* **have** *5*: A **by** (*rule ccontr*)
    from *5* **have** *6*: $A \vee \neg A$ **by** (*rule disjI1*)
    from *1 6* **have** *False* **by** (*rule notE*)
  }
  from *this* **have** $A \vee \neg A$ **by** (*rule ccontr*)
  thus *?thesis* **.**
qed

1.6 Exercise 1c: logical expressions natural language3

prop $\exists\, ship.\ huge(ship) \wedge blue(ship)$
prop $\neg shining() \longrightarrow sad(me)$
prop $raining \vee not\text{-}raining$
prop $going(she) \longrightarrow going(me) \wedge \neg going(she) \longrightarrow \neg going(me)$
prop $\forall\, x.\ loves\text{-}chocholate(x) \vee loves\text{-}icecream(x)$
prop $\exists\, x.\ loves\text{-}icecream(x) \wedge loves\text{-}chocolate(x)$
prop $\forall\, x.\exists\, y.\ play\text{-}together(x,\ y)$
prop $\forall\, x.\ isMean(x) \longrightarrow (\forall\, x.\ \forall\, y.\ \neg play\text{-}together(x,\ y))$
prop $\forall\, p.\ isAnnoying(p) \wedge ofDog(p) \longrightarrow ofCat(p)$

2 Exercise 2

2.1 a)

theorem *A*:
  assumes *1*: $A \wedge B \longrightarrow C$
  assumes *2*: $B \longrightarrow A$
  assumes *3*: B
  shows C
proof −
  from *2 3* **have** *4*: A **by** (*rule mp*)
  from *4 3* **have** *5*: $A \wedge B$ **by** (*rule conjI*)
  from *1 5* **have** *6*: C **by** (*rule mp*)
  thus *?thesis*.
qed

2.2 b)

theorem *B*:
  assumes *1*: A

```
  shows B ⟶ A
proof -
  {
    assume B
    from 1 have A by -
  } note 2 = this
  from 2 have 3: B ⟶ A by (rule impI)
  thus ?thesis.
qed
```

2.3 c)

```
theorem C:
  assumes 1: A⟶(B⟶C)
  shows B ⟶ (A ⟶C)
proof -
  {
    assume 2: B
    {
     assume 3: A
     from 1 3 have 4: B⟶C  by (rule mp)
     from 4 2 have 5: C by (rule mp)
    }
    from this have A ⟶ C by (rule impI)
  }
  from this have B ⟶ (A ⟶ C) by (rule impI)
  thus ?thesis.
qed
```

2.4 d)

```
theorem D:
  assumes 1: ¬A
  shows A ⟶ B
proof -
  {
    assume 2: A
    {
     assume 3: ¬B
     from 1 have 4: ¬A by -
     from 2 have 5: A by -
     from 4 5 have False by (rule notE)
    }
    from this have ¬¬B by (rule notI)
    from this have B by (rule notnotD)
  }
  from this have A ⟶ B by (rule impI)
  thus ?thesis.
qed
```

2.5 e)

```
theorem E:
  assumes 1: ¬A
  shows A ⟶ B
proof -
  {
    assume 2: A
    {
     assume 3: ¬B
     from 1 have 4: ¬A by -
     from 2 have 5: A by -
     from 4 5 have False by (rule notE)
    }
    from this have ¬¬B by (rule notI)
    from this have B by (rule notnotD)
  }
  from this have A ⟶ B by (rule impI)
  thus ?thesis.
qed
```

3 A Hilbert Proof Calculus for Propositional Logic (PL)

3.1 Logical Connectives for PL

3.1.1 Primitive Connectives

```
consts impl :: bool ⇒ bool ⇒ bool (infixr → 49)
consts not :: bool ⇒ bool (¬)
```

In philosophy, we often assume that the only two logical connectives are the implication *op* $\rightarrow$ and the negation $\neg$. This is handy, since it simplifies proofs to only consider these two cases.

3.1.2 Further Defined Connectives

We can of course add further connectives that are to be understood as abbreviations that are defined in terms of the primitive connectives above.

```
abbreviation disj :: bool ⇒ bool ⇒ bool (infixr ∨50) where
  A ∨ B ≡ ¬A → B
abbreviation conj :: bool ⇒ bool ⇒ bool (infixr ∧51) where
  A ∧ B ≡ ¬(A → ¬B)
```

3.2 Hilbert Axioms for PL

3.2.1 Axiom Schemes

axiomatization where

A2: $A \to (B \to A)$ **and**
A3: $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ **and**
A4: $(\neg A \to \neg B) \to (B \to A)$

3.2.2 Inference Rules

axiomatization where
ModusPonens: $(A \to B) \Longrightarrow A \Longrightarrow B$

lemma *True* **nitpick** [*satisfy*, *user-axioms*, *expect* = *genuine*] **oops**

3.3 A Proof

thm *A3*[**where** $A = A$ **and** $B = (B \to A)$ **and** $C = A$]
thm *A3*[*of* A $(B \to A)$ A]

We show that A1 is redundant

theorem *A1Redundant*:
shows $A \to A$
proof –
have *1*: $(A \to ((B \to A) \to A)) \to ((A \to (B \to A)) \to (A \to A))$ **by** (*rule* *A3*[**where** $B = (B \to A)$ **and** $C = A$])
have *2*: $A \to ((B \to A) \to A)$ **by** (*rule* *A2*[**where** $B = B \to A$])
from *1 2* **have** *3*: $(A \to (B \to A)) \to (A \to A)$ **by** (*rule ModusPonens*)
have *4*: $(A \to (B \to A))$ **by** (*rule A2*)
from *3 4* **have** *5*: $A \to A$ **by** (*rule ModusPonens*)
thus *?thesis* **.**
qed

theorem
shows $A \to A$
by (*metis* (*full-types*) *A2 ModusPonens*) — Sledgehammer even finds a proof without using A3

4 Exercise 3

theorem *exercise3*:
assumes *1*: $A \to B$
assumes *2*: $B \to C$
shows $A \to C$
proof –

have *3*: $((B \to C) \to A \to (B \to C))$ **by** (*rule A2*[**where** $A = B \to C$ **and** $B = A$])
from *3 2* **have** *4*: $A \to (B \to C)$ **by** (*rule ModusPonens*)
have *5*: $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ **by** (*rule A3*)
from *5 4* **have** *6*: $((A \to B) \to (A \to C))$ **by** (*rule ModusPonens*)
from *6 1* **have** $A \to C$ **by** (*rule ModusPonens*)
thus *?thesis*.
qed